DOCUMENT RESUME

ED 349 964 IR 015 703

AUTHOR Katz, Sandra; Lesgold, Alan
TITLE Use of Fuzzy Modeling Techniques in a Coached Practice Environment for Electronics Troubleshooting.
PUB DATE May 92
NOTE 27p.; Paper presented at the Annual Conference of the National Council on Educational Measurement (San Francisco, CA, April 1992).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Artificial Intelligence; *Computer Assisted Instruction; Computer Simulation; Electronics; Knowledge Level; *Models; Problem Solving; *Programed Tutoring; Technical Education; *Troubleshooting
IDENTIFIERS Coached Practice; *Fuzzy Set Theory; *Intelligent Tutoring Systems

ABSTRACT

Student modeling--the task of building dynamic models of student ability--is fraught with uncertainty, caused by such factors as multiple sources of student errors, careless errors and lucky guesses, learning and forgetting. Within the context of the Sherlock intelligent tutoring systems project, we have been experimenting with various ways of making the task of modeling student knowledge more tractable. The philosophical basis underlying each approach is that student models do not need to be precise and accurate to be useful. We describe these approaches, focusing on the one we have developed furthest thus far. The approach, which is based on fuzzy set theory, aims at building imprecise, or "fuzzy" diagnostic student models (e.g., Hawkes et al., 1990). We have built upon this approach by developing techniques for representing and updating discrete student knowledge variables in our avionics troubleshooting tutor, Sherlock II. We describe these techniques and, more broadly, the student modeling component in this tutor. We frame our discussion of the "fuzzy" student modeling approach we are developing with a description of its more crude predecessor, and of our plans for future work on imprecise student modeling using Bayesian inferencing techniques. (Contains 52 references.) (Author/BBM)

# Use of Fuzzy Modeling Techniques in a Coached Practice Environment for Electronics Troubleshooting

*Sandra Katz and Alan Lesgold*
*Learning Research and Development Center*
*University of Pittsburgh*
*katz@unix.cis.pitt.edu*

Draft of May 28, 1992

Paper presented at the National Council on Educational Measurement's Annual Conference
San Francisco, California

# Use of Fuzzy Modeling Techniques in a Coached Practice Environment for Electronics Troubleshooting


*Abstract*

Student modeling—the task of building dynamic models of student ability—is fraught with uncertainty, caused by such factors as multiple sources of student errors, careless errors and lucky guesses, learning and forgetting. Within the context of the **Sherlock** intelligent tutoring systems project, we have been experimenting with various ways of making the task of modeling student knowledge more tractable. The philosophical basis underlying each approach is that student models do not need to be precise and accurate to be useful. We describe these approaches, focusing on the one we have developed furthest thus far. The approach, which is based on fuzzy set theory, aims at building *imprecise*, or "fuzzy" diagnostic student models (e.g., Hawkes et al., 1990). We have built upon this approach by developing techniques for representing and updating discrete student knowledge variables in our avionics troubleshooting tutor, **Sherlock II**. We describe these techniques and, more broadly, the student modeling component in this tutor. We frame our discussion of the "fuzzy" student modeling approach we are developing with a description of its more crude predecessor, and of our plans for future work on imprecise student modeling using Bayesian inferencing techniques.


*Introduction*

The **Sherlock** tutors—**Sherlock I** (Lajoie and Lesgold, 1990; Lesgold et al., 1990) and **Sherlock II** (Katz et al., in press; Lesgold et al., in press)—exemplify a form of job-situated training that we call a *coached practice environment*. A coached practice environment provides students with opportunities to practice problem-solving activities in a realistic setting, with computer-supplied support (coaching) available when needed. The automated coach also gives the student feedback on his or her strengths and weaknesses, and assigns new problems driven by this assessment. The subject domain of the **Sherlock** tutors is fault diagnosis in the Manual Avionics Test Station, a complex electronic testing system that is used to check modules of F-15 aircraft.

In order to make decisions about the *amount* of help a student needs at different stages of instruction, what the *content* of that help should be, which tasks a student is ready to handle, etc., a tutor—human or machine—needs to construct a mental representation of the student's competency in the domain being taught. However, deriving these models is difficult, because a student's actions are the sole "window" a tutor has for observing a student's ability, and the process of interpreting these actions is fraught with uncertainty. Ambiguity is one source of uncertainty; there is often more than one possible explanation for students' errors and inappropriate actions. Multiplicity is another; an error or inappropriate problem-solving action can often be traced to several misconceptions and skill deficiencies. Factor in idiosyncratic errors such as computational or mechanical slip-ups (e.g., typos), lucky guesses, the fact that students often forget prior knowledge, etc. and it is no wonder that Self (1990) has pronounced the task, "the intractable problem of student modeling."

Several approaches to making student modeling more tractable have been developed in

recent years. These approaches vary across a spectrum of precision in the models produced.[1] At one extreme lie model-tracing methods (e.g., Anderson, Boyle, & Reiser, 1985) and buggy diagnostic approaches (e.g. VanLehn, 1988), which are often combined. With model-tracing technology, the knowledge that the student is expected to acquire is represented within the tutor as a set of executable production rules. Common buggy productions also may be included in the model. Student performance is matched to expected performance based on the expert model. By restricting the student to a particular problem space, it is technically possible for the system to understand exactly what the student is doing using these model-tracing techniques. However, model-tracing methods tend to eliminate trial-and-error search and exploration, thereby reducing opportunities for metacognitive skill development, and for system modeling of metacognitive skill (Derry & Hawkes, 1992).

Examples of less precision-oriented modeling approaches include *bounded student models* (Elsom-Cooke, 1989) and *granularity-based recognition* of students' problem-solving plans and strategies (e.g., Greer et al., 1989; McCalla, Greer, et al., 1992). These approaches are grounded in the belief that student models do not need to be precise and accurate to be useful. Similarly, in developing the **Sherlock** tutors, we have been experimenting with various ways of building imprecise, dynamic models of student ability on discrete knowledge components. This paper focuses on the main approach we have implemented to date. The approach is grounded in fuzzy set theory (Zadeh, 1965), which attempts to capture the notion that items can have varying *degrees* of membership in a set, as opposed to the standard view that an item either belongs or does not belong in a set. For example, a student might have partial membership within the set of people who are expert in a particular skill, as reflected in teacher comments such as, "Joe is *fairly good* at two-column subtraction." We frame our discussion of the "fuzzy" student modeling approach we are developing with a description of its more crude predecessor, and of our plans for future work on imprecise student modeling using Bayesian inferencing techniques.

## *Building Imprecise Student Models*

### Where We Started: Competence/Performance Modeling in Sherlock I

**Sherlock I** was an initial "proof of concept" of the *coached practice environment* design idea.[2] As such, it contained a partial device simulation that was rather limited, and a crude though surprisingly effective student modeling scheme. For each problem, **Sherlock I** contained an and/or goal-structure tree. The top-level goal, solving the problem, might be satisfied, in whole or in part, through several different approaches. In general, any given goal in the tree could be satisfied by one of possibly several approaches (the "or" aspect of the tree), each of which might contain several subgoals (the "and" aspect of the tree).

A student's performance could then be characterized in terms of how good his moves were at each subgoal node of the goal structure that his performance "touched." We used only

---

[1]See the Introduction to Lajoie and Derry (in press) for a more in-depth discussion of this issue.

[2]The Air Force evaluation of **Sherlock I** found that subjects who spent twenty to twenty-five hours working with the system were as proficient in troubleshooting electronic faults in the test station as technicians who had been on the job four years (Nichols et al., in press). More detailed discussions of **Sherlock I** can be found in (Lajoie & Lesgold, 1989; Lesgold & Lajoie, 1990; Lesgold et al., in press).

three performance ratings: *Good, Ok,* and *Bad*, reflecting expert, acceptable non-expert, and wrong moves respectively. We hand coded, for each node of the goal tree for each separate problem, a list of knowledge components we thought were needed to attain the given goal. For each student, we maintained a student *competence model* which was simply a rating of competence for each knowledge component on our list. Following Anderson's (1983) ACT model, we used four rating categories for knowledge components: *Unlearned, Perhaps, Probably,* and *Strong*. This is similar to the overlay modeling approach, except that it allows for a range of qualitative ratings, rather than a simple boolean distinction—i.e., the student has/has not mastered a knowledge component.

When the student began a problem, a predicted *performance model* would be generated by estimating how well the student might do on each subgoal of the goal tree. The estimate for a given node was essentially an average of the ratings of the student on the knowledge components listed for that node. After the student completed the problem, the actual performance was compared node by node with the expected performance, and the ratings for knowledge components associated with a node were adjusted to reflect the student's actual performance. Upgrading towards *Strong* was done conservatively, only in light of repeated evidence, so that Sherlock would not overestimate the student's performance on future problems.

The main use of the student model in **Sherlock I** was to tailor advice when the student asked for help. Apparently, Sherlock's ability to predict how much help a student needed right away worked quite well despite the crudeness of the underlying scheme. It was unusual for students to ask for more than one or two hints at a given subgoal node (Lajoie & Lesgold, 1989). However, the main limitations of the approach were: (1) from a knowledge engineering standpoint, the burden of building a separate goal tree for each problem, and associating knowledge components with each subgoal node, and (2) the vagueness of the competency estimates (*Unlearned, Perhaps,* etc.), which are inadequate to give a student feedback on his or her performance, or to serve other system functions that depend on at least some level of cognitive diagnosis.

## "Fuzzy" Student Modeling in Sherlock II

Several improvements were embodied in **Sherlock II** (Lesgold et al., in press), one of which was deeper intelligence, mainly in the device (manual test station) modeling capability and the expert modeling capability. Both forms of deep intelligence paved the way for richer, more intelligent student modeling in **Sherlock II**. In this section, we briefly describe **Sherlock II**, and the main approach to imprecise modeling we have implemented within it thus far.

*Overview of the tutor.* **Sherlock II** is a realistic computer simulation of the F-15 maintenance job environment. Trainees acquire and practice skills in a context similar to the real setting in which they will be used. The tutor, Sherlock, presents trainees with a series of exercises of increasing difficulty. There are two main episodes in each exercise: *problem-solving* and *review*. During problem-solving, the student runs a set of checkout procedures on an aircraft module suspected of malfunction (known as the "unit under test," or UUT), which is automatically attached to the test station. Using interactive video with a mouse pointer interface, the student can set switches and adjust knobs and dials on test station drawers, take measurements, and view readings. If he gets an unexpected reading on one of the measurement devices (handheld meter, digital multimeter, or oscilloscope), he should see if the aircraft module is the "culprit" by replacing it with a shop standard. If after doing this the student still gets an unexpected reading, he should troubleshoot the test station. He can test components by "attaching" probes to measurement points in a video display, replace a suspect component with a shop standard and

rerun the failed checkout test, etc. Perhaps most importantly, the student can ask for advice at any point while troubleshooting. Sherlock provides advice at both the circuit path and individual component levels of investigation. These and other options are available through a menu-driven interface, as shown in Figure 1.

In designing **Sherlock II**, we embraced Vygotsky's (1978) notion of cognitive tools: "objects provided by the learning environment that permit students to incorporate new auxiliary methods or symbols into their problem-solving activity that otherwise would be unavailable" (Derry & Hawkes, 1992; Lajoie & Derry, in press). The major cognitive tool provided by **Sherlock II** is an intelligent graphics interface that helps students construct a mental model of the circuitry involved in carrying out a failed checkout test, and keep track of the status of troubleshooting goals. A sample abstract schematic diagram is shown in Figure 2. Although not visible here, these diagrams are color-coded to indicate which parts of the circuitry have been found from previous tests to be good (green); which areas are bad (red); and which are not suspect (yellow) or of unknown status (black). The drawings are interactive: mousing on any component box produces an explanation of what is known about the status of that component given the actions (such as measurements and swaps) carried out so far. An intelligent schematic producer configures these drawings to match the coach's current explanatory goal, the current problem solving context, and information about the trainee. For example, more space and more expanded circuit detail are provided in the part of the circuit on which an expert might now be focusing or the part the coach wishes to emphasize. Component labeling and color coding are filtered to assure that diagrams don't unintentionally give away too much information.[3]

Each problem-solving session is followed by a review phase, which we call *reflective follow-up* (RFU) Psychological experimentation (Owen & Sweller, 1985; Sweller, 1988; Sweller & Cooper, 1985) and theoretical models of case-based learning (e.g., Mitchell, Keller, & Kedar-Cabelli, 1986) indicate why a review phase is important for acquiring cognitive skills. Students often suffer from "cognitive overload" during problem-soiving sessions. Consequently, it is best to parcel out some of the instruction to a post-problem reflective phase. The following options are available during reflective follow-up : (a) *Show Sherlock's comments on my solution*, (b) *Replay my solution*, (c) *Replay Sherlock's solution*, (d) *Compare my solution to Sherlock's*, (e) *Summarize my progress through the Sherlock program*, (f) *Explain what the test station was supposed to be doing at the point of failure in the problem just finished*, and (g) *Let me help determine the next problem Sherlock assigns*. Menu labels are actually more terse than described here, as shown in Figure 3.

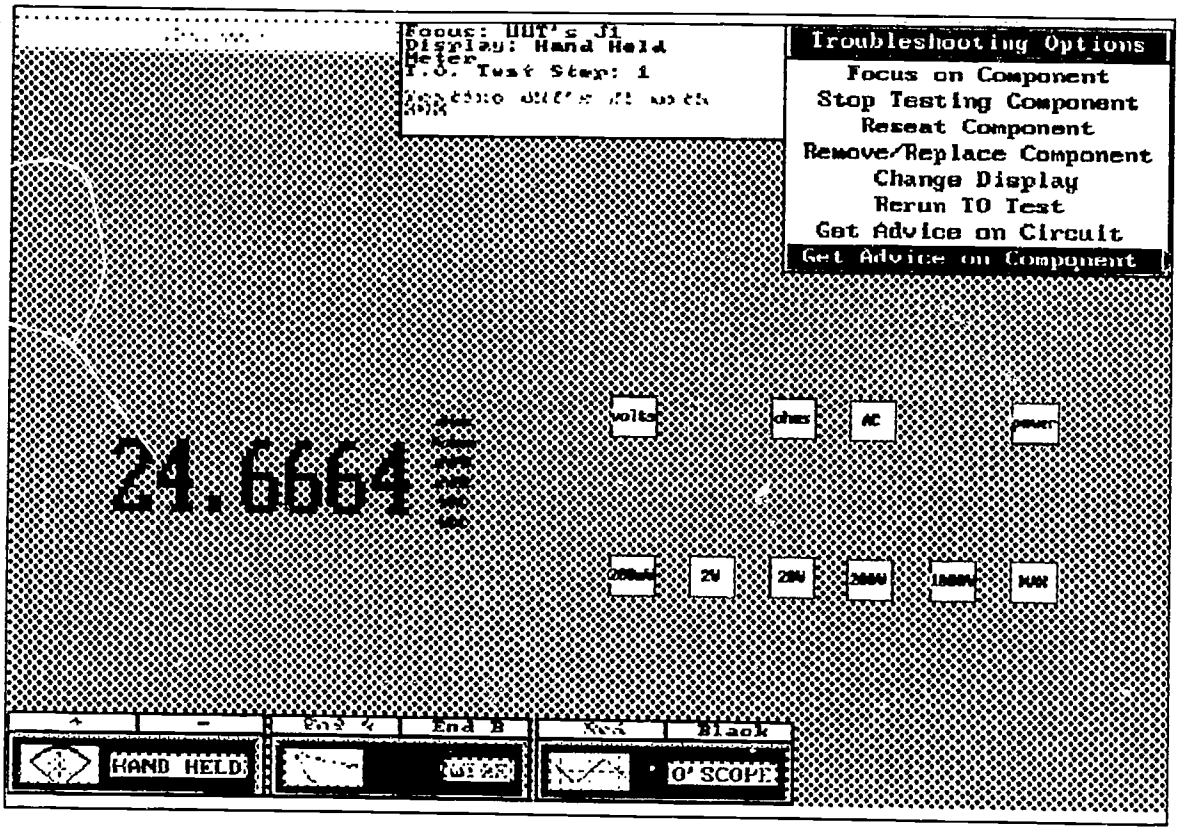
It is beyond the scope of this paper to describe each menu option, but we will give the reader a sense of what the main review activity we want students to engage in is like: namely, a coached replay of the student's problem-solving session. While stepping through his solution within the *Replay my solution* option, the student receives a summary of each action and feedback pointing out its good and bad aspects. A **Sherlock II** comment is illustrated in Figure 4.[4] The student can ask to see what Sherlock would have done instead, and receive other kinds of coaching, including the chance to re-examine the dynamically labeled and color-

---

[3]A component is only labeled if the student has already tested it, if it has been referred to in a help message, or if the student has identified the component when prompted. This restriction ensures that students have to use schematics to trace through the active path.

[4]Johanna Moore is currently developing tools to generate more intelligent explanations during reflective follow-up, drawing from ideas in her dissertation (Moore, 1989).

Figure 1: Sherlock's Troubleshooting Menu



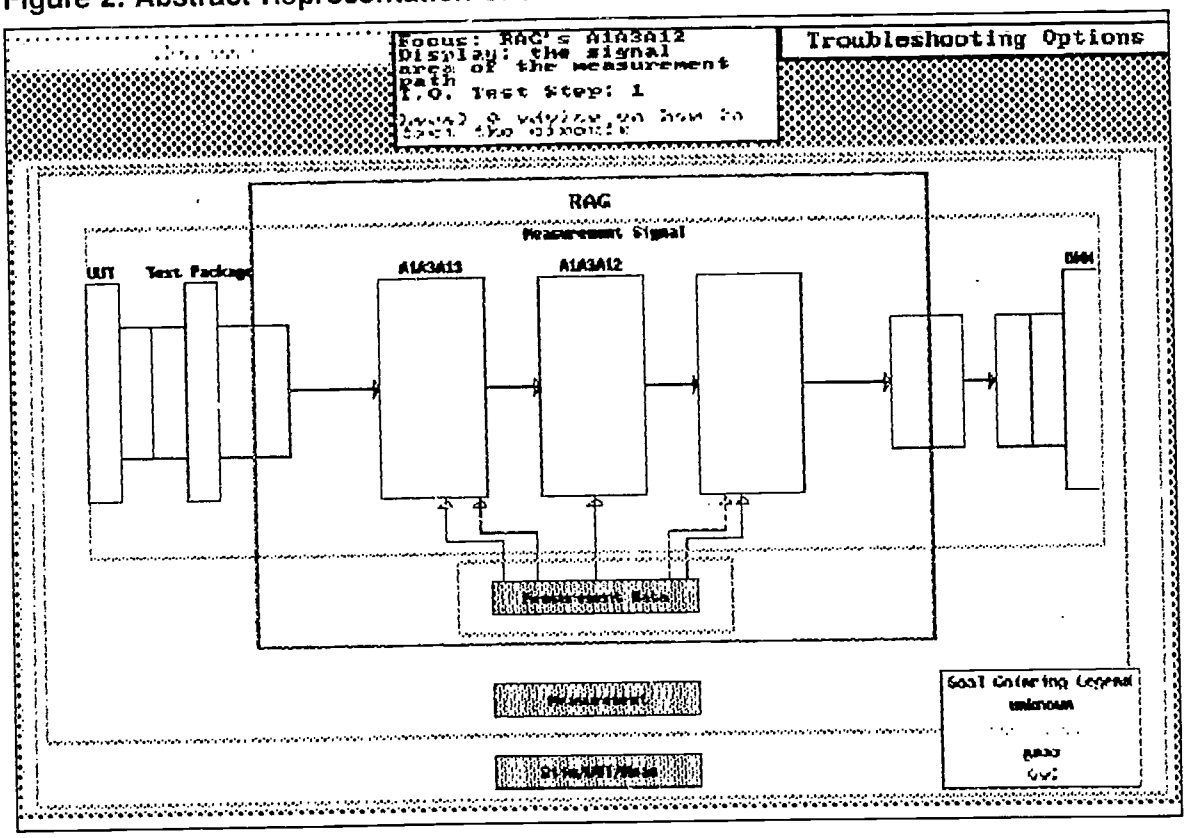

coded schematic diagrams visible during problem-solving. (See Figure 2.)

*Motivation for experimenting with "fuzzy" modeling techniques.* "Fuzzy" (imprecise) student modeling was originally proposed by Hawkes, Derry and their colleagues (e.g., Hawkes et al., 1990; Derry & Hawkes, 1992). Hawkes et al. (1990) present the following rationale for applying fuzzy set theory to student modeling:

> [Partial membership within a set] is an important concept to the field of ITSs [intelligent tutor systems] because there are different aspects of vagueness inherent in real world data. There is the inherent vagueness of classification terms referring to a continuous scale, the uncertainty of linguistic terms such as "I almost agree," or the vagueness of terms or concepts due to statistical variability in communication (Zemankova & Kandel. 1984). Fuzzy set theory is an attempt to provide a systematic and sound basis for the modeling of [those] types of imprecision which are mainly due to a lack of well-defined boundaries for elements belonging to the set of objects. The use of fuzzy terms [eg., "rather high," "possibly," "not likely," etc.] allows for imprecision and vagueness in the values stored in the database. This provides a flexible and realistic representation that easily captures the way in which the human tutor might evaluate a student...Also, many tutoring decisions are not clearcut ones and the capability to deal with such imprecision is a definite enhancement to ITS's. (pp. 416-17)

We have built upon Hawkes' and Derry's work by developing techniques for representing and updating discrete "fuzzy" student knowledge variables. The basic scheme can be described as follows. In **Sherlock II**, each knowledge variable is associated with a *fuzzy probability distribution* that is upgraded or downgraded at different rates depending upon the type and strength of the evidence that appears in a student problem-solving trace. For example, the variable *ability to interpret test results* receives a strong upgrade each time a student tests the input signals to a circuit card when a previous test shows that the card's output signals are faulty, but receives a weaker upgrade if the student performs the input verification after receiving system

Figure 2: Abstract Representation of the Active Circuit Path

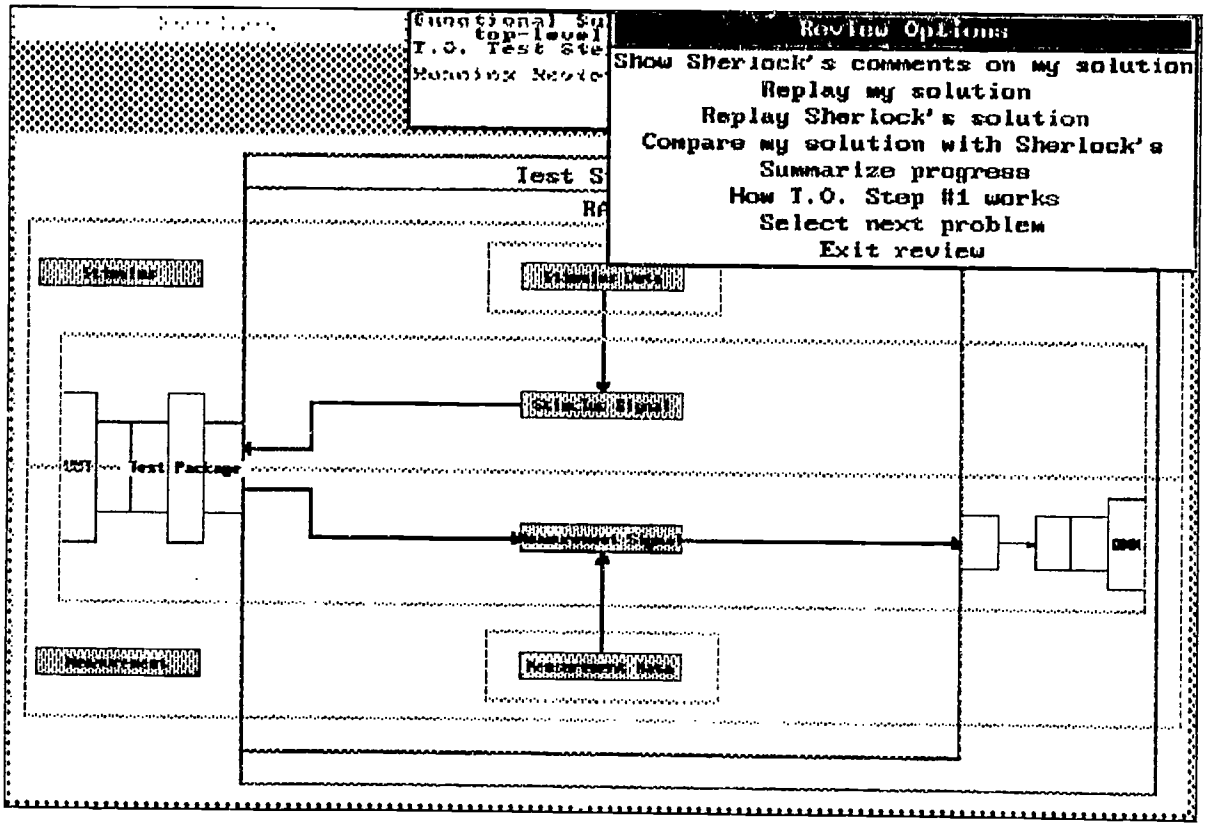



advice to do so. Knowledge variables are linked together in a lattice where higher-order variables, which we call "global variables" (e.g., *ability to use test equipment*) represent aggregations of more primitive knowledge components, which we call "local variables" (e.g., skill in using each type of test equipment—i.e., *ability to use the handheld meter*; *ability to use the digital multimeter*; *ability to use the oscilloscope*). The updating techniques we have developed propagate changes in local variables "upwards" through layers of associated global variables.

A major technical aim of our work has been to simplify the knowledge and software engineering requirements involved in building a student modeling component that can produce fuzzy (imprecise) probabilistic student records which are adequate to guide system functions such as coaching, feedback, and problem selection. There is a tradition of formal probabilistic approaches to reasoning under uncertainty in the field of AI, most notably *Bayesian inferencing networks*[5] and an outgrowth of fuzzy set theory known as *possibility theory* (e.g., Zadeh, 1965, 1975, 1978, 1986; Dubois & Prade, 1979, 1987). However, some approaches to possibility theory have been criticized for overwhelming computational complexity, while simpler ones compromise expressibility (Schwartz, 1989). And Bayesian reasoning, although computationally manageable (e.g., Pearl, 1988), has often been criticized for high knowledge engineering demands (e.g.,

[5]For a thorough technical discussion of Bayesian belief networks see Pearl (1988). A more accessible introduction can be found in Charniak (1991) or Morawski (1988).

Figure 3: Reflective Follow-up Options




Villano, 1992).[6] That is, knowledge engineers (often domain experts) face the difficult and unnatural task of specifying prior probabilities for variables and conditional probabilities for link matrices.

In contrast, our modeling scheme allows system developers to state in broad, categorical terms how indicative a particular action is of a particular knowledge attribute (e.g., *strong*, *medium*, *weak*). In addition, as the following pages will demonstrate, the techniques we use to update primitive knowledge variables and draw inferences from these primitives about higher-order knowledge variables are far simpler computationally than those used in either Bayesian or fuzzy reasoning systems. A long-term goal of our research is to determine just how much simplicity and imprecision a student modeling engine can get away with and still be useful.

*The Student Modeling Knowledge Base: A Network of "Fuzzy Variables."* While the student is working on a problem, the system records a trace of his actions, including the components he tested; the actual measurements made; replacements ("swaps") of components with shop standards; requests for help, etc. This *student trace* is the crucial input to the student modeling component, which interprets the trace in order to update the tutor's assessment of the student's knowledge state on particular modeling variables.

---

[6]There is also a long-standing debate about the relative adequacy of fuzzy and Bayesian approaches for representing and reasoning about vague concepts. It is beyond the scope of this paper to discuss this issue, but we refer the interested reader to Cheeseman (1986); Cohen (1985); Schwartz (1989); and Zadeh (1986).

Figure 4: Sherlock's Comments on a Troubleshooting Action

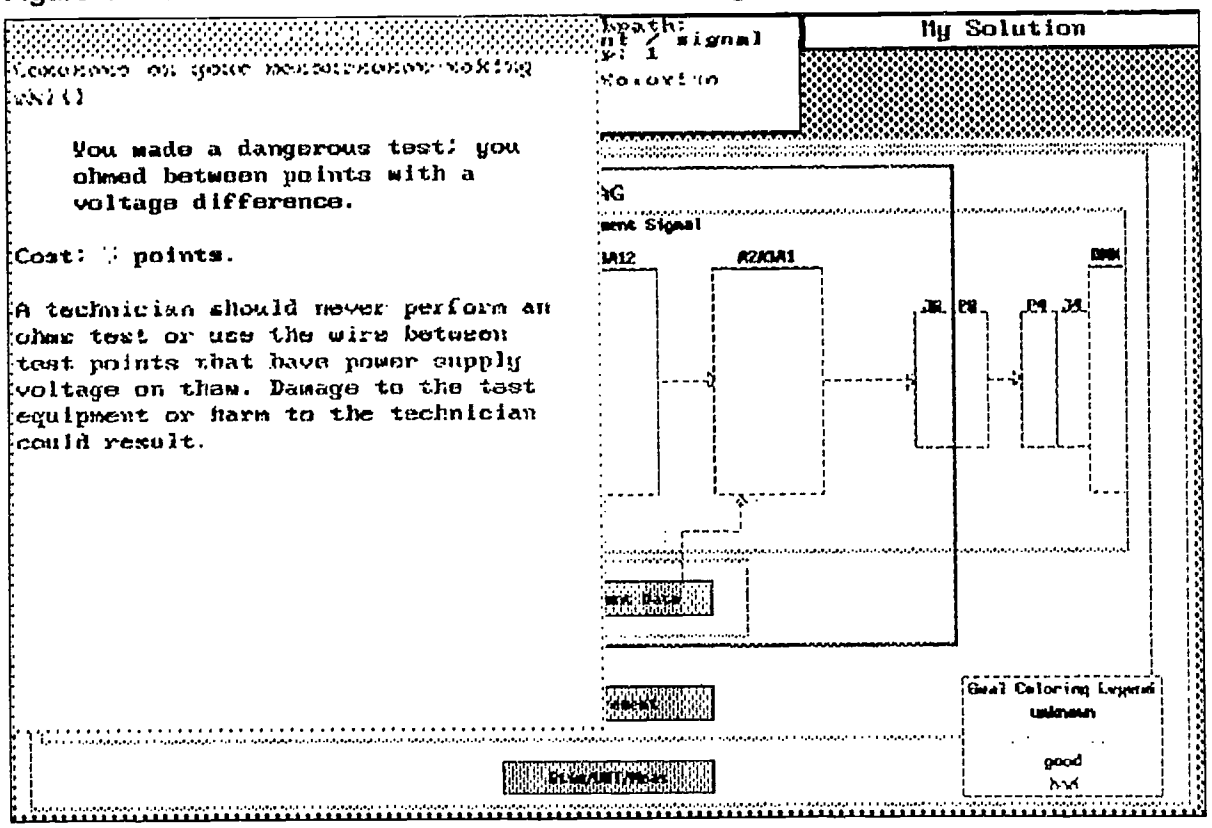
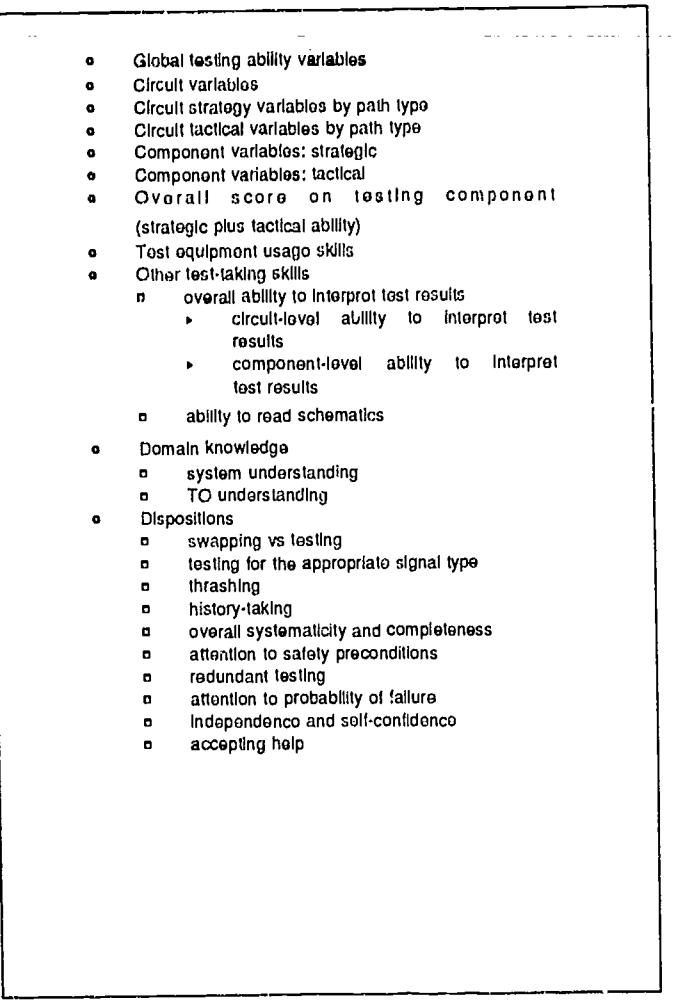



Figure 5 contains a list of the main types of variables tracked by **Sherlock II**'s student modeling component.[7] Where do these modeling variables come from? We relied upon two main sources of information about which aspects of students' understanding and performance should be modeled by the system: cognitive task analysis, and expert judgments. Local variables, and the rules for updating them, were derived mainly from observable properties of troubleshooting performance, as revealed during cognitive analysis of the job of troubleshooting the F-15 manual test station (Lesgold et al., 1990; Lesgold & Lajoie, 1990). To a lesser degree, local variables also reflect domain experts' judgments about what properties of local performance are important to measure. Global variables, on the other hand, are anchored primarily in expert evaluations of trainee performance. Policy-capturing techniques helped us to identify the evaluation criteria experts use to rate student performance traces (Nichols et al., in press).[8] Indirectly, global variables are also anchored in observations of human performance, since they represent cumulations of local variables. These relationships between modeling variables and

[7]The most indented items under each category are local variables; the rest are global.

[8]The set of evaluation criteria identified by using policy-capturing techniques keeps the amount of sensible cumulations of local variables (i.e., global variables) to a tractable size.

**Figure 5: Types of Modeling Variables in Sherlock II**




their sources are summarized in Figure 6.[9]

Currently, all student modeling variables are represented as *fuzzy variables*. A fuzzy variable can be thought of as a distribution over the set of possible levels of competence (i.e., knowledge states) a trainee might have in a particular skill, or understanding of a particular domain concept. **Sherlock II** tracks five such knowledge states for each modeling variable: *no knowledge, limited knowledge, unautomated knowledge, partially automated knowledge,* and *fully developed knowledge.* The distribution of a fuzzy variable can be denoted by the vector $F$, with the $i$th probability interval being denoted by $F_i$.

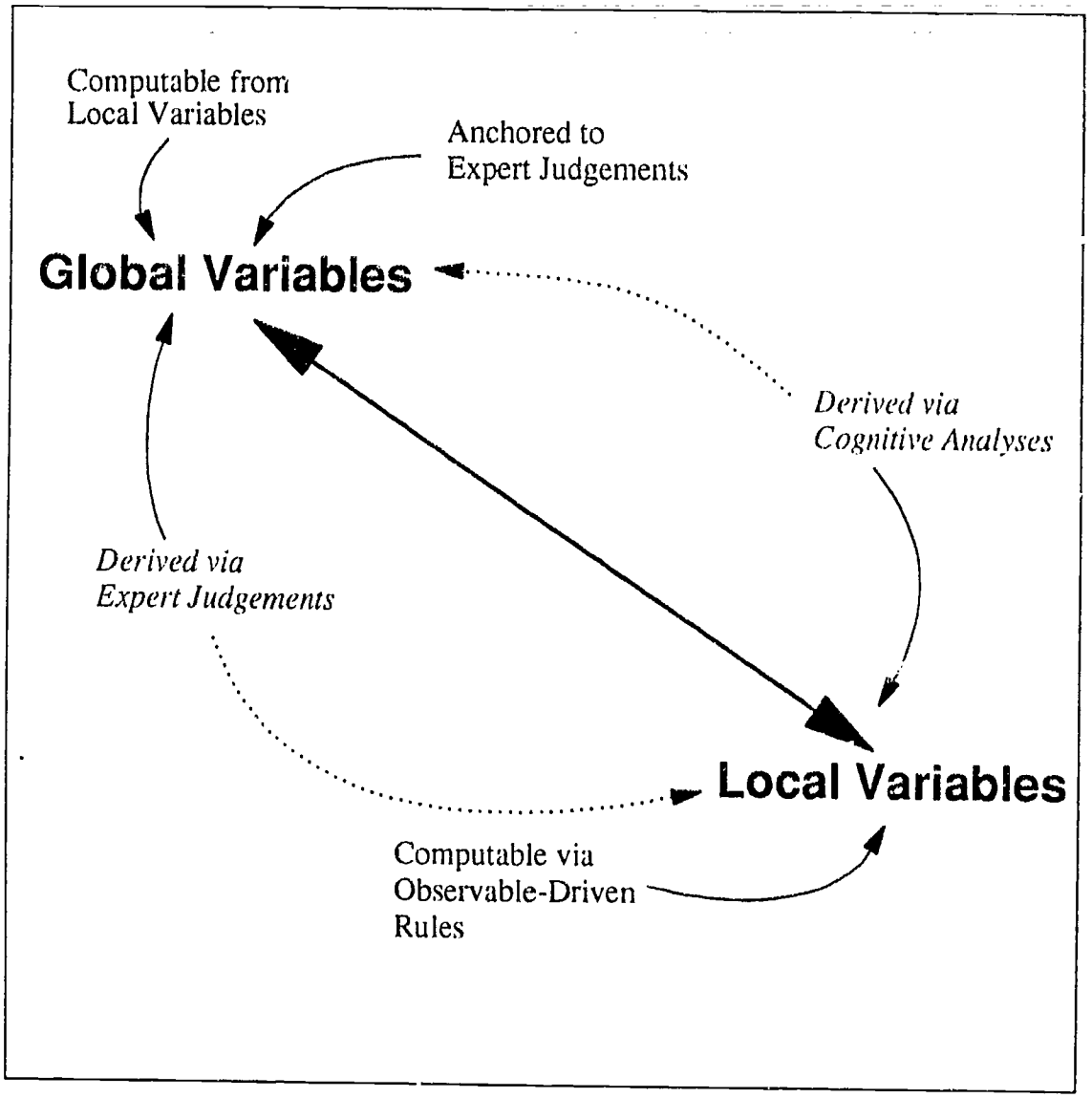
For most knowledge variables, we assume that each of the five states has equal probability (20%), and we initialize the distributions accordingly, as illustrated in Figure 7.[10] However, since we have some prior knowledge about student ability on certain variables, we can bias these distributions. For example, since we know that most of our students have had little experience using an oscilloscope, we can initialize the associated distribution as (20 60 20 0 0). This would indicate that we are 60% certain that the skill indexed by this particular variable is limited in any given student, but it might be non-existent, and it might even have reached the level of being established but not automated. The updating procedures described below control revisions of these initial hypotheses about student ability.

*Variable Updating: Local Variables.* Everything that a teacher—human or machine—can infer about a student's knowledge and misconceptions is conveyed through the student's actions, be they speech acts or non-linguistic actions. Indeed, no user modeling system can peer directly into an agent's mind; language and action are the sole media through which modeling information passes, and there will invariably be gaps and distortions in the image conveyed. This is, in essence, why the user modeling task is so hard. In a system like **Sherlock II**, which does not

[9]Solid lines represent primary sources of information about variables; dotted lines represent secondary sources.

[10]More rigorous analysis of student data is needed to more accurately initialize our modeling variables.

**Figure 6. Multi-Level Assessment Scheme.**




have natural-language processing capabilities,[11] student actions serve as the sole "window" or "keyhole" (Cohen, Perrault & Allen, 1982) through which diagnostic information about the student is conveyed. **Sherlock II**'s modeling window is the *student trace*.

Of course, only a subset of student actions observable by a computerized tutor are

[11]As noted in footnote 5, Johanna Moore is currently extending the dialogue capabilities within **Sherlock II**'s reflective follow-up phase. One goal of her work is to enable trainees to ask follow-up questions about the comments that the system generates while replaying the student's (or an expert's) solution. Eventually, similar dialogue capabilities will be incorporated within **Sherlock II**'s problem-solving phase. Kass (1990) has commented upon the importance of flexible interaction between the system and the user for building a user model.

significant, from a diagnostic standpoint. In our tutoring system, we refer to these 'diagnostically significant' aspects of student behavior as *performance conditions*. Performance conditions comprise the "lefthand side" of rules for updating local variables, in conventional terms: the events that trigger the system to update the current distribution for a local fuzzy variable. In fact, we view local variables as abstractions over performance conditions, just as global variables are meaningful abstractions over local variables.

The performance conditions that update the fuzzy variable representing a student's ability to use a handheld meter are shown in Figure 7. Adjacent to each condition is a symbol indicating how rapidly the variable should be updated when the rule fires. We use an arbitrary convention in this example, where +++ means to upgrade the variable relatively quickly, ++ means a moderate upgrade, + means a slower upgrade, --- means a rapid downgrade, etc. Below, we formalize the procedure that carries out modifications of fuzzy distributions at these varying rates.

There are two main observations to be made about performance conditions. First, although the updating rules shown in this example are derived from simple observations (e.g., what test type the student set the meter to, voltage or ohms), not all local variables need to be fixed by simple events. Rather, some may have their values inferred by relatively complex qualitative patterns over multiple observable events. For instance, the local variable for the student's strategy in testing a particular functional area of the test station (e.g., the signal path between the UUT and the digital multimeter) may operate on implications of actions, such as an index of the extent to which a *series* of measurements on components within this functional area helped to identify the location of the fault.[12]

Perhaps a more important observation to be made about performance conditions is that they are redundant across variables. This is not surprising, since in reality there could be more than one valid explanation for a student action. An action can be part of different plans and/or motivated by different aspects of conceptual understanding.

To take an example with reference to the handheld meter (Figure 7), an incorrect setting on this measurement device—e.g., ohms instead of voltage—could mean that the student does not realize that setting the meter before testing is necessary, or does not know how to do this, from a purely mechanical standpoint. This aspect of domain knowledge is tracked by the variable *tactical test equipment skill*. What is more likely, though, is that the student does not know that the correct setting for the test he is conducting is VDC (direct current voltage), because he does not realize that current was in the active circuit path for the diagnostic procedure that the test station was carrying out when failure occurred. In other words, the student probably has an inadequate mental model of the failed checkout test.[13]

---

[12]See Lesgold et al. (in press) for additional examples of performance conditions used to update local variables.

[13]This situation provides a good example of how violations of expected behavior can serve as indicators of gaps in students' knowledge. Kass (1990) stipulated a general rule for acquiring information about an agent called the Sufficiency Rule. The rule is based upon Grice's principles of cooperative interaction. It essentially states that a cooperative agent does everything necessary (and nothing more) to enable the system to achieve its goal. In a tutorial situation, it is usually the agent, not the system, who has the goal of solving the problem, but the Sufficiency Rule still applies to a large extent. That is, we can expect that the student will do what is required to solve the problem (e.g., find the fault in a test station's active circuitry), barring some satisficing. If he does not, it is probably because of a lack of understanding, or lack of capacity

Figure 7: A Sample Fuzzy Variable

ABILITY TO USE THE HANDHELD METER

**description:** measure of knowledge about when to use the handheld meter; tactical ability is assumed

**initial distribution:** (20 20 20 20 20)

**upgrading rules:**

1) uses handheld meter appropriately to measure resistance, i.e., when there is no power, without help from Sherlock (+++)
2) uses handheld meter appropriately to measure DC voltage, i.e., DC power is on, without help from Sherlock (+++)
3) uses handheld meter appropriately to measure AC voltage, i.e., AC power is on (+++)

**downgrading rules:**

1) uses handheld meter to measure resistance when power is on (e.g., shorting in the data area) (---)
2) uses handheld meter to measure voltage when there is none (---)
3) uses handheld meter to measure DC voltage when it's AC (--)
4) uses handheld meter to measure AC voltage when it's DC (--)

In our system, we handle the ambiguity in interpreting student actions by allowing the same performance condition to update *several* variables. Regarding our example, the act of performing the wrong type of test will update *tactical test equipment skill*, *ability to use the handheld meter*, *testing for the appropriate signal type*, etc. However, we vary the rate at which updating occurs for a given condition across variables. Two factors govern the update rate that our knowledge engineers assign to a performance condition for a particular variable: (1) the condition's strength as an indicator of competency in that variable, and (2) the frequency with which the action associated with the condition might occur during any given problem-solving session. If the same upgrading speed were used for all events, some variables would reach expert level way ahead of others. For example, there are many more opportunities to set the handheld meter, to place probes on pins, and to carry out other test-taking actions on a component than there are to respond to an indicator light which fails to come on.

Identifying updating rates that take both factors (indicative strength of a behavior, and its frequency of occurrence) into account is indeed a tricky matter at times. There are two clear-cut cases: weak indicators that occur frequently should update a variable slowly; strong indicators that occur infrequently should update a variable quickly. We generally handle the other cases, in which the indicative strength of a condition and its frequency of occurrence are at odds, by

---

due to insufficiently automated knowledge.

assigning a moderate updating rate. As with all parameters in our modeling system, these rates need to be adjusted empirically.

The updating rate is expressed in the "righthand" side of an updating rule—the "action" part, in conventional terms. More formally, we specify the updating rate for a fuzzy variable, *F*, by two pieces of information: a *range vector*, *V*, and a *change percentage*, *c*. The change percentage controls the rate of updating. The downgrading procedure for a fuzzy variable, *F*, can be expressed as follows, where *V* represents the range vector[14] and *c* the change percentage:

$F = (f_1\ f_2\ f_3\ f_4\ f_5)$,
$V = (v_1\ v_2\ v_3\ v_4\ v_5)$,
$c$ = constant,
$f_i = f_i - f_i v_i c + f_{i+1} v_{i+1} c$, where $v_1 = 0$, $i = 1,...4$,
$f_5 = f_5 - f_5 v_5 c$

An example is shown in Figure 8. The basic idea behind downgrading is to "shift" the current distribution to the left, by successively adding some amount of probability interval $F_{i+1}$ to $F_i$. The leftmost interval in the change vector is set to 0, to prevent "spillover." As the example shows, downgrading proceeds conservatively, since a single instance of an error is insufficient for drawing conclusions about student ability. However, as Figure 8 shows, multiple behavioral events *are* significant, and effect dramatic changes in the distribution.

Upgrading occurs in a similar manner except that the shift is to the right, by successively adding some amount of $F_i$ to $F_{i+1}$. The rightmost interval in the change vector is set to 0, again to prevent spillover:

$f_1 = f_1 - f_1 v_1 c$,

$f_i = f_i - f_i v_i c + f_{i-1} v_{i-1} c$, where $v_5 = 0$, $i = 2,...,5$

Given the same initial values for *F*, *V*, and *c* shown in Figure 8, a single upgrade would change *F* to (18 20 20 21.4 20.6). After ten rule firings, the distribution would be (6.3 13.9 18.2 32.9 28.7).

*Variable Updating: Global Variables.* Reasoning under uncertainty can be viewed as a two-sided operation. One involves *pooling* evidence to arrive at hypotheses; the other involves *propagating* the uncertainty in one or more hypothesis through an inferencing process aimed at arriving at some conclusion (Cohen, 1985). In the preceding section, we described how our system pools behavioral evidence (performance conditions) to formulate hypotheses about student competence on "primitive" (local) skills. In this section, we describe how Sherlock combines these hypotheses to reason about more complex, "global" skills.

In **Sherlock II**, updating rules for global variables are expressed as weighted linear

[14] In our system, we vary the range according to the student's current level of expertise—*novice*, *journeyman*, or *near-expert*. The percentage for each interval in the range decreases as skill increases. For example, the range for a journeyman might be (0, 20, 30, 30, 30), for a near-expert (0, 15, 25, 25, 25). This scheme enables us to control updating so that the system does not deem the student an "expert" too readily.

equations. Just as the change rates in updating rules for local variables reflect a performance condition's strength as an indicator for that variable, so do the weights in aggregation equations reflect the relative strength of each local knowledge variable in determining the student's ability on the associated global knowledge variable. Below are some sample weighted equations for updating global variables.

**Circuit testing ability** = .85 *circuit testing strategy* + .15 *circuit testing tactical ability*

**Test equipment usage** = .60 *ability to use the oscilloscope* + .20 *ability to use the digital multimeter* + .20 *ability to use the handheld meter*

**Domain understanding** = .75 *system understanding* + .25 *understanding of the checkout procedures*

The result of applying these equations is a composite distribution, expressing the system's degree of belief that the student is competent in the skill represented by the global variable. The procedure for updating a global variable $G$, using local variables $F_i$ and their associated weights, $w_i$, can be expressed as shown below.

$$G = w_1F_1 + w_2F_2 + \ldots + w_kF_k,$$

$$F_1 = (f_{11}\, f_{12}\, f_{13}\, f_{14}\, f_{15})$$
$$F_2 = (f_{21}\, f_{22}\, f_{23}\, f_{24}\, f_{25})$$
$$\ldots$$
$$F_k = (f_{k1}\, f_{k2}\, f_{k3}\, f_{k4}\, f_{k5})$$

We use the following formula to calculate $G = (g_1\ g_2\ g_3\ g_4\ g_5)$:

$$g_1 = w_1f_{11} + w_2f_{21} + \ldots + w_kf_{k1},$$
$$\ldots$$
$$g_5 = w_1f_{15} + w_2f_{25} + \ldots + w_kf_{k5}$$

An example for *test equipment usage*, where the number of associated local variables, $k$, equals 3 is shown in Figure 9.

**Figure 8: Example of Variable Downgrading**

**Initial settings:**

$F = (20\ 20\ 20\ 20\ 20)$,
$V = (0\ 30\ 100\ 100\ 100)$,
$c = 10$

**After one downgrade:**

$f_i = f_i - f_iv_ic + f_{i+1}v_{i+1}c$, where $v_1 = 0$, $i = 1,\ldots4$
$f_5 = f_5 - f_5v_5c$

$f_1 = f_1 - f_1v_1c + f_2v_2c$
$= 20 - 20 \times 0 \times .1 + 20 \times .3 \times .1 = 20 - 0 + .6 = 20.6$

$f_2 = f_2 - f_2v_2c + f_3v_3c$
$= 20 - 20 \times .30 \times .1 + 20 \times 1 \times .1 = 20 - .6 = 2 = 21.4$

$f_3 = f_3 - f_3v_3c + f_4v_4c$
$= 20 - 20 \times 1 \times .1 + 20 \times 1 \times .1 = 20 - 20 + 20 = 20$

$f_4 = f_4 - f_4v_4c + f_5v_5c$
$= 20 - 20 \times 1 \times .1 + 20 \times 1 \times .1 = 20 - 20 + 20 = 20$

$f_5 = f_5 - f_5v_5c$
$= 20 - 20 \times 1 \times .1 = 20 - 2 = 18$

$F = (20.6\ 21.4\ 20\ 20\ 18)$

**After 10 downgrades:**

$F = (28.7\ 32.9\ 18.2\ 13.9\ 6.3)$

*Putting it All Together: Updating an Individual Student Model.* The "intelligence" in object-oriented systems like **Sherlock II** is encapsulated in computational "objects." An object is an independent piece of computer program that stores its own local data and can thus respond to various requests that other parts of the system might make of it. In **Sherlock II**, the information needed to model the student gets recorded by objects at two levels:

**Figure 9: Example of Global Variable Updating**

*G - test equipment usage*

| | |
|---|---|
| $F_1$ - *ability to use the oscilloscope* | $F_1$ = (0 40 30 30 0) |
| $F_2$ - *ability to use the digital multimeter* | $F_2$ = (0 20 60 20 0) |
| $F_3$ - *ability to use the handheld meter* | $F_3$ = (0 0 30 50 20) |

$G = .6F1 + .2F_2 + .2F_3$

$G = (g_1\ g_2\ g_3\ g_4\ g_5)$
$g_1 = .6 \times 0 + .2 \times 0 + .2 \times 0 = 0$
$g_2 = .6 \times 40 + .2 \times 20 + .2 \times 0 = 28$
$g_3 = .6 \times 30 + .2 \times 60 + .2 \times 30 = 26$
$g_4 = .6 \times 30 + .2 \times 20 + .2 \times 50 = 32$
$g_5 = .6 \times 0 + .2 \times 0 + .2 \times 20 = 4$

$G = (0\ 28\ 36\ 32\ 4)$

the active circuit path, and individual component objects within this path. The circuit path object records global information such as the order of components tested, and the order in which circuit testing goals (e.g., verification that the signal path between the UUT and the digital multimeter is functioning ok) were achieved. Component objects record more local information about what happened while the student was testing a particular component—in particular, the order in which tests were made and component testing goals (e.g., verification that the inputs to the component are ok) were achieved; which particular pins were measured and how the probes were placed; what test equipment was used, and how it was set up; what types and levels of coaching were requested, etc. The modeling information gathered by these circuit and component objects is stored in the object that we call the 'student trace.'[15] Consequently, the student competency model can be stored in a separate data structure (object) than the knowledge required for device and expert simulation, so it can be used across problems, rather than hand-coded for each problem as it was in **Sherlock I**.

A few variables are updated dynamically, while the student is solving a problem. However, most variables are updated between problem completion and reflective follow-up, since updating takes time and would slow down the system considerably if it were all done dynamically. Dynamically updated variables primarily correspond to safety hazards, such as attempting to conduct an ohms test when current is on. Dynamic updating is triggered by rule "firings," as soon as these dangerous events occur. The appropriate variables are updated the rate specified by their updating rules—in particular, the "righthand" side which encodes the updating rate.

---

[15]Actually, the student trace consists of a *cluster* of objects, each one holding a different type of information—e.g., one records the student's measurements, one holds the sequence of achieved goals and the tests used to achieve these goals, another stores information about test device settings, etc. This division of labor makes updating the model more efficient, since individual updating routines need only access those objects that contain the information that they need. However, for conceptual simplicity and ease of exposition, it is best to think in terms of one *student trace* object.

Post-solution updating proceeds as follows. First, Sherlock imports the student's record, thereby gaining access to the student's score on all local variables up to the current exercise. If the student is new to the system, Sherlock initializes variables to a pre-specified level, as indicated by the "initial distribution" slot in Figure 7. Using its updating routines, Sherlock examines the student's solution trace, searching for performance conditions that trigger rule firings. Each performance condition is recorded in a structure called the *conditions table*. Sherlock simply notes how many times each performance condition has been identified in the student trace. Each local variable associated with that indicator will then be updated the recorded number of times. After each local variable has been updated, Sherlock uses its weighed linear equations to propagate these values "upwards" through the student modeling lattice, thereby updating global variables.

## *Current Status and Future Directions*

### An Interlude: "Cost Scoring" in a Later Version of **Sherlock II**

A prototype version of **Sherlock II** underwent initial field trials in August of 1991. The system contained an uncalibrated student modeling component, implemented using the approach described in this article. These initial system trials underscored the need to fine-tune the student modeling knowledge base in light of student data. We found, for example, that student records were being updated too slowly to be useful for problem selection. In general, students were much better than the modeler made them out to be. This was one of a cluster of reasons why some students were given too many easy problems and became frustrated with the tutor.[16] Our plans for calibrating the student modeling lattice and updating rules are discussed in (Katz & Lesgold, 1991; Katz et al., in press).

We expect such fine-tuning to take considerable effort, and more student data than was made available during the preliminary field trials. Since we were in need of a more empirically valid student modeling/assessment component in order to demonstrate and field test the tutor in the summer of 1992, we implemented an interim scheme based on the results of an empirically validated study of expert scoring criteria (Pokorny & Gott, in press). This study revealed the competency indicators (behaviors) that experts watch for, the high-level ("global") skills they associate with these behaviors—in particular, *understanding of the testing system (test station)*, *test-taking ability*, and *strategy*—and the relative import experts attach to a particular behavior as evidence of a particular skill. The resulting variable lattice is quite shallow, revealing only coarse-grained associations between student behaviors and a small set of global variables.

The modeling approach we applied to this shallow network is based on a more

---

[16]Other factors contributing to sluggish curriculum planning included a problem set that was too large, and a non-robust file backup procedure which caused some student trace files to be deleted. The latter led the tutor to consider students who had already worked on the tutor beginners!

traditional, point-scoring scheme, in which students are "charged" a certain number of points for inappropriate as well as some appropriate troubleshooting actions (e.g., extending a circuit card for testing; replacing a malfunctioning component). Their total charge is then compared with that incurred by the expert model run on the same problem. The ratio of expert-to-student "costs" constitutes the student's score for a particular problem, and then this score is used to update a recency-weighted average of the student's performance on particular problem types (e.g., problems in which a switch is malfunctioning; problems that require use of the oscilloscope). This information is then used to guide selection of the next problem. If the ratio is over a certain threshold, the student will be advanced a certain number of problems, which are ordered by increasing difficulty; if not, he will be moved down or kept within the same range.[17] In addition, a shallow clustering of inappropriate actions under global variable labels (i.e., *system understanding*, *test-taking ability*, and *strategy*) is presented to the student for feedback purposes. The strengths of this approach are (1) the simplicity of its way of dealing with uncertainty, (2) the transparency of the scoring facets for feedback, and (3) the opportunity the approach affords to let student self-assessments contribute to student modeling. Uncertainty is handled by always interpreting student performance in relation to expert performance, instead of in absolute terms. Transparency of feedback is achieved by focusing on actual behaviors (e.g., "You set up the handheld meter for ohms, when a voltage setting was necessary.") rather than on abstract curriculum goals (e.g., *ability to use the handheld meter*). The basic scheme we have designed for enabling student self-assessments to contribute to student modeling can be sketched as follows:

— Show the student his location in the problem difficulty continua.
— Let him move the current placement indicator up or down.
— Give him a problem at the level requested.
— Decide where he should have been placed based on his performance (either further "up" or further "down").
— If he did well, take his word that he is better than the system "thinks," and move his placement indicator up by some amount (between the current placement and student placement).
— If not, don't overcorrect the placement indicator, since the student was clearly in over his head.
— Propagate the "corrected" placement level backward through the aggregation network.

In effect, this scheme revives the central idea behind the student modeling approach taken in **Sherlock I**: to update the student's competency model in response to discrepancies between expected performance and actual performance. The idea of back-propagating scores through the variable network stems from connectionist modeling, but we expect that it can be done using Bayesian inferencing techniques, which we discuss below.

This summer's field trials will enable us to assess the effectiveness of this rather standard assessment approach. The approach itself will also serve as a useful basis for future comparisons of a calibrated version of the more experimental, "fuzzy" approach to student modeling and assessment with the Bayesian approach we are starting to develop.

[17]See Katz et al. (1992) for a more detailed discussion of this approach to problem selection.

## Future Directions: Application of Bayesian Inferencing Techniques to Student Modeling

In the near future, we will build a Bayesian inferencing network from a calibrated version of the student modeling network described in this paper. A long-term goal is to compare the effectiveness of using standard Bayesian updating algorithms for student modeling with those we have developed for the "fuzzy" and "cost scoring" approaches.

There are several motivations for experimenting with a Bayesian student modeling approach. The variable lattice we have developed is already quite close to a Bayesian belief net. Its nodes represent discrete knowledge variables, encoded as distributions over meaningful knowledge states (*unautomated, partially automated*, etc.). However, the current network structure will have to be checked to ensure that it adheres to the constraint of *conditional independence*, which prevents updating inertia by ensuring that all of the nodes that a given node depends upon are linked directly to it (Pearl, 1988). We might find that Bayesian inferencing techniques do a better job than ours at updating the student's knowledge state on variables within the revised network.

One possible reason for this is that unlike our updating scheme, where reasoning takes place in only one direction, Bayesian belief nets are quite capable of *bi-directional reasoning*. This has been demonstrated extensively in other domains, especially medicine. That is, in addition to formulating diagnostic hypotheses, a Bayesian reasoning system such as MUNIN (Andreassen et al., 1987) can make *predictions* about the occurrence of symptoms as these hypotheses are strengthened or weakened, which can then be used to guide the selection of further tests. Similarly, a bi-directional, Bayesian student modeling scheme might enable the system to reason about student ability on knowledge variables for which little or no evidence has been gathered, as well as predict student performance on particular types of problems. Another motivation for experimenting with Bayesian student modeling is that belief networks are already a standard AI technique. As such, they are understood and taught, which increases the likelihood that Bayesian student modeling components will be implemented in other tutors. This would in turn lead to the development of a well-codified, standard technology for Bayesian student modeling. Several researchers have already begun to experiment with this approach (e.g., Villano, 1992). We hope to be able to say more about our plans for carrying out Bayesian student modeling during the workshop.

Our future comparisons of the three modeling approaches described above—imprecise (fuzzy) modeling, the point-scoring scheme, and Bayesian modeling—will focus on several issues, mainly:

(1) the types of tutoring system functions each approach is best-suited for

(2) the amount of knowledge engineering effort required to implement an initial version of a given approach

(3) the amount of knowledge engineering effort required to calibrate the modeling component

(4) the difficulty of programming the updating procedures

It may well be, for example, that Bayesian student modeling is superior to our current fuzzy modeling scheme for problem selection, because of the greater mathematical rigour of Bayesian belief nets, and their bi-directional reasoning capability. However, this

increased functionality might have to be bought at greater costs in terms of knowledge engineering and programming effort.

The difference in engineering effort might not be as great as we originally expected, however. In the first place, our fuzzy modeling component turned out to require more effort to prototype than we anticipated, and we expect the work required to calibrate the many parameter settings (e.g., updating rates, and the weights in aggregation equations) to be substantial. If calibration remains difficult, then the marginal cost of implementing a Bayesian scheme could be low. Recent work in test theory provides some rigorous computational methods for deriving probability estimates (e.g., Mislevy, in press; Bock & Aitkin, 1981; Lewis, 1985; Tsutakaw & Johnson, 1990) for Bayesian belief nets. However, *calibrating* these estimates might prove to be unwieldy, due to feedback loops. That is, it could be nearly impossible to assign blame correctly when estimated student knowledge states in the network fail to match human expert ratings. For example, if variable *B*, which depends upon variable *A*, appears to be inaccurate, is it because *A*'s prior probabilities are wrong, or is it because *B* at some point made an incorrect prediction about *A*, which then "backfired" (propagated upwards) to *B*? Our future experiments with both modeling approaches will enable us to address these knowledge engineering complexity issues.

## *Summary and Conclusions*

We have described our experiments with various approaches to imprecise student modeling. Our central aim so far has been to further the development of an approach to imprecise modeling of student ability based upon fuzzy set theory (e.g., Derry & Hawkes, 1992; Hawkes et al., 1990). Technical concerns included minimizing the knowledge engineering effort required to initially develop and fine-tune the student modeling knowledge base, while also lessening computational (programming) complexity. Our main contribution to date has been in specifying and implementing relatively simple procedures that can dynamically update fuzzy (imprecise) probability distributions which represent student competence on discrete knowledge components. The adequacy of these procedures will be determined when the student modeling knowledge base and updating routines in **Sherlock II** have been calibrated, and the tutoring system is evaluated.

Several researchers who have been experimenting with imprecise modeling approaches are finding that the incomplete and inaccurate models produced are nonetheless useful for carrying out the system's knowledge assessment and didactic functions (e.g., Chin, 1989; Derry & Hawkes, 1992; Greer & McCalla, 1989; McCalla & Greer, 1992). In our own work on the forerunner of **Sherlock II** (**Sherlock I**), we found that a very crude categorization of student ability into discrete knowledge levels (i.e., *unlearned*, *perhaps*, *probably*, and *strong*) worked quite well in guiding system decisions about the level of detail to provide in hints (Lesgold et al., in press). This is not surprising, since there is an increasing body of evidence that human tutoring decisions seldom involve precise diagnosis (e.g., McArthur, Stasz, & Zmuidzinas, 1990; Putnam, 1987)—that is, a detailed model of the misconceptions or "bugs" that motivate student

errors—although research has not yet directly addressed the issue of exactly what kinds of student models expert human tutors in action do construct. We share with Derry and Hawkes (1992) the belief that in low-risk decision-making situations such as tutoring, where new information is constantly being made available for modifying diagnostic hypotheses, imprecise student modeling is adequate. Future evaluations of our system and others incorporating imprecise modeling approaches will help to determine how effective these approaches actually are.

An evaluation of a particular approach should include comparisons with other student modeling approaches. Such comparisons should focus on the knowledge engineering effort needed to develop an initial version of the student modeling component within a given approach, and then to fine-tune it; the difficulty of implementing the approach; and, perhaps most importantly, how useful the approach is for different system functions (i.e., coaching, versus curriculum planning, versus assessment, etc.). Other researchers (e.g., Cahour & Paris, 1991; Kay, 1991) have stressed the importance of designing user modeling components by taking into account situational factors such as what the model will be used for, what the available human resources are, and what the developmental stage of the system (i.e., prototype, or near-release) is. We believe that empirically validated comparisons of various approaches to student modeling—addressing functionality and engineering complexity issues such as those listed above—will better equip tutoring system designers to take these situational factors into account, and thereby make more informed decisions about which modeling approach (or combination of approaches) to take in their tutor.

The work on **Sherlock II** described in this article is funded by the Air Force. The funding agency and the collaborators acknowledged below do not necessarily endorse the views expressed.

*Acknowledgements*

**Sherlock II** has been a collaborative effort by a team that has included (either currently or in the recent past) Marilyn Bunzo, Richard Eastman, Gary Eggan, Maria Gordin, Linda Greenberg, Edward Hughes, Sandra Katz, Susanne Lajoie, Alan Lesgold, Thomas McGinnis, Rudianto Prabowo, Govinda Rao, and Rose Rosenfeld. As with its predecessor, **Sherlock I**, Dr. Sherrie Gott and her colleagues at Air Force Human Resources, Armstrong Laboratories, are active contributors to the effort. We would especially like to thank Linda Greenberg for help with preparing the manuscript and Sharon Derry for many informative discussions about the work on fuzzy diagnosis and assessment that she has been doing with Lois Hawkes and her colleagues at Florida State University.

*References*


Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science, 228*, pp. 456-468.

Andreassen, S., Woldbye, M., Falck, & Andersen, S.K. (1987). MUNIN: a causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the 10th International Joint Conference on Artificial Intelligence.*

Bock, R.D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: an application of an EM-algorithm. *Psychometrika,* 46, 443-459.

Cahour, B., & Paris, C. (1991). Role and use of user models. In J. Kay and A. Quilici (Eds.), *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence Workshop W.4: Agent Modelling for Intelligent Interaction,* 73-80.

Charniak, E. (1991). Bayesian networks without tears. *AI Magazine.* 12 (4), 50-63.

Cheeseman, P. (1986). Probabilistic versus fuzzy reasoning. In L.N. Kanal & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence (pp. 85-102).* North-Holland: Ensevier Science Publishers.

Chin, D.N. (1989). KNOME: modeling what the user knows in UC. In A. Kobsa & W. Wahlster (Eds.), *User models in dialog systems (pp. 74-107).* New York: Springer-Verlag.

Cohen, P.R. (1985). *Heuristic reasoning about uncertainty (pp. 14-48).* Boston: Pitman Advanced Pub. Program.

Cohen, P.R., Perrault, C.R., & Allen, J.F. (1982). Beyond question answering. In W.G. Lehnert & M. Ringle (Eds.). *Strategies for natural language processing (pp. 245-74).* Hillsdale, NJ: Lawrence Erlbaum Associates.

Derry, S.J., & Hawkes, L.W. (1992). Toward fuzzy diagnostic assessment of metacognitive knowledged and growth. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

DuBois, D., & Prade, H. (1979). *Fuzzy sets and systems: theory and applications.* New York: Academic Press.

DuBois, D. & Prade, H. (1987). Necessity measures and the resolution principle. *IEEE Transactions on Systems, Man and Cybernetics,* 17, 474-78.

Elsom-Cook, M. (1989). Guided discovery tutoring. Preprint for Nato Workshop on Guided Discovery Tutoring, Italy.

Greer, J.E., & McCalla, G.I. (1989). A computational framework for granularity and its application to educational diagnosis. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (Vol. 1, pp. 477-82).*

Greer, J.E., McCalla, G.I., & Mark, M.A. (1989). Incorporating granularity-based recognition into SCENT. *Proceedings of the Fourth International Conference on Artificial Intelligence and Education (pp. 107-15),* Amsterdam.

Hawkes, L.W., Derry, S.J., & Rundensteiner, E.A. (1990). Individualized tutoring using an intelligent fuzzy temporal relational database. *International Journal of Man-Machine Studies,* 33, 409-429.

Kass, R. (1990). Building a user model implicitly from a cooperative advisory dialog. *Proceedings of the Second International Workshop on User Modeling,* Honolulu, Hawaii.

Katz, S., Lesgold, A., Eggan, G., & Gordin, M. (in press). Modeling the student in **Sherlock II.** To appear in the *Journal of Artificial Intelligence in Education (special issue on student modeling, edited by G.I. McCalla and J. Greer).*

Katz, S., & Lesgold, A. (1991). Modeling the student in **Sherlock II.** In J. Kay & A. Quilici (Eds.), *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence Workshop W.4: Agent Modelling for Intelligent Interaction (pp. 73-80),* Sydney, Australia.

Kay, J. (1991). Generalised user modelling shells—a taxonomy. In J. Kay and A. Quilici (Eds.), *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence Workshop W.4: Agent Modelling for Intelligent Interaction (pp. 169-85).*

Lajoie, S.P., & Derry, S.J. (in press). Introduction. In S.P. Lajoie & S. Derry (Eds.), *Computers as cognitive tools.* NJ: Lawrence Erlbaum Associates.

Lajoie, S.P. & Lesgold, A. (1990). Apprenticeship training in the workplace: computer-coached practice environments as a new form of apprenticeship. *Machine-Mediated Learning,* 3, 7-28.

Lesgold, A.M. & Lajoie, S.P. (1990). Complex problem solving in electronics. In R. Sternberg & P.A. Frensch (Eds.), *Complex problem solving: principles and mechanisms.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A.M., Lajoie, S.P., Logan, D., & Eggan, G. (1990). Applying cognitive task analysis and research methods to assessment. In N. Frederiksen, R. Glaser, A.M. Lesgold, & M. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition (pp. 325-50).*

Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A.M., Eggan, G., Katz, S., & Rao, G. (in press). Possibilities for assessment using computer-based apprenticeship environments. To appear in W. Regian & V. Shute (Eds.), *Cognitive approaches to automated instruction.* Hillsdale, NJ: Erlbaum.

Lewis, C. (1985). Estimating individual abilities with imperfectly known item response function. Paper presented at the Annual Meeting of the Psychometric Society, Nashville TN.

McArthur, D., Stasz, C., & Zmuidzinas, M. (1990). Tutoring techniques in algebra. Cognition and Instruction, 7(3), 197-244.

McCalla, G., & Greer, J.E. (1992). Enhancing the robustness of model-based recognition. Paper presented at the Third International Workshop on User Modeling, Dagstuhl Castle, Germany.

McCalla, G.I., Greer, J.E., Barrie, B., & Pospisil, P. (1992). Granularity hierarchies. To appear in *International Journal of Computers and Mathematics with Applications (Special Issue on Semantic Networks).*

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: a unifying view. *Machine Learning,* 1, 47-80.

Mislevy, R.J. (in press). Randomization-based inference about latent variables from complex samples. *Psychometrika.*

Moore, J.D. (1989). A reactive approach to explanation in expert and advice-giving systems. *PhD thesis, University of California, Los Angeles.*

Morawski, P. (1989). Understanding Bayesian belief networks. *AI Expert,* May, 44-48.

Nichols, P., Pokorny, R., Jones, G., Gott, S.P., & Alley, W.E. (in press). *Evaluation of an avionics troubleshooting tutoring system.* Special Report. Brooks Air Force Base, TX: Air Force Human Resources Laboratory.

Owen, E., & Sweller, J. (1985). What do students learn while solving mathematics problems? *Journal of Educational Psychology,* 77(3), 272-284.

Pearl, J. (1988). Evidential reasoning under uncertainty. In H. E. Shrobe and the American Association for Artificial Intelligence (Eds.), *Exploring artificial intelligence: survey talks from the National Conferences on Artificial Intelligence (pp. 381-418).* San Mateo: Morgan Kaufmann Publishers, Inc.

Pokorny, B., & Gott, S.P. (in press). The evaluation of a real-world instructional system: using technical experts as raters. Technical Report. Brooks Air Force Base: Air Force Human Resources.

Putnam, R.T. (1987). Structuring and adjusting content for students: a study of live and simulated tutoring of addition. *American Educational Research Journal*, 24(1), 13-48.

Schwartz, D. (1989). Outline of a naive semantics for reasoning with qualitative linguistic information. *Proceedings of IJCAI-89*, 20-25.

Self, J.A. (1990). Bypassing the intractable problem of student modeling. In C. Frasson and G. Gauthier (Eds.), *Intelligent tutoring systems: at the crossroad of artificial intelligence and education* (pp. 107-123). Norwood, NJ: Ablex Publishing Corporation.

Sweller, J. (1988). Cognitive load during problem solving: effects on learning. *Cognitive Science*, 12, 257-85.

Sweller, J. & Cooper, G.A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1), 59-89.

Tsutakawa, R.K., & Johnson, J. (1990). The effect of uncertainty of item parameter estimation on ability estimates. *Psychometrika*, 55, 371-90.

VanLehn, K. (1988). Toward a theory of impasse-driven learning. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems (pp. 19-42)*. New York: Springer-Verlag.

Villano, M. (1992). Probabalistic student models: Bayesian belief networks and knowledge space theory. Paper presented at the Second International Conference on Intelligent Tutoring Systems, Montreal.

Vygotsky, L.S. (1978). *Mind in society*. Cambridge, MA: Harvard University Press.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8, 338-353.

Zadeh, L.A. (1975). The concept of a linguistic variable and its application to approximate reasoning, Part I: *Information Science*, 8, 199-249; Part II: *Information Science*, 8, 301-357; Part III: *Information Science*, 9, 43-80.

Zadeh, L.A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, vol. 1, No. 1, 3-28.

Zadeh, L.A. (1986). Outline of a theory of usuality based on fuzzy logic. In A. Jones, A. Kaufmann, & H-J. Zimmerman (Eds.), *Fuzzy sets: theory and applications*, Dordrecht:

Reidel, 79-97.

Zemankova-Leech, M. & Kandel, A. (1984). *Fuzzy Relational Data Bases—a Key to Expert Systems*, Koeln: Verlag TNV Rheinland.